# The Notion of Variability in Software Architecture – Results from a Preliminary Exploratory Study

Galster, Matthias; Avgeriou, Paris

*Published in:*
EPRINTS-BOOK-TITLE



*Publication date:*
2011

# The Notion of Variability in Software Architecture – Results from a Preliminary Exploratory Study

Matthias Galster
Software Engineering and Architecture Group
Department of Mathematics and Computer Science
University of Groningen
The Netherlands
m.r.galster@rug.nl

Paris Avgeriou
Software Engineering and Architecture Group
Department of Mathematics and Computer Science
University of Groningen
The Netherlands
paris@cs.rug.nl

## ABSTRACT
*Context:* In the software product line domain, the concept of variability is well recognized. However, variability in the context of software architecture still seems to be poorly understood. *Objective:* In this paper, we aim at contributing to the development of a basic understanding of the notion of variability in the software architecture domain, beyond the idea of product lines. *Method:* We perform a preliminary exploratory study which consists of two parts: an expert survey among 11 subjects, and a mini focus group with 4 participants. For both parts, we collect and analyze mostly qualitative data. *Results:* Our observations indicate that there seems to be no common understanding of "variability" in the context of software architecture. On the other hand, some challenges related to variability in software architecture are similar to challenges identified in the product line domain. *Conclusions:* Variability in software architecture might require more theoretical foundations in order to establish "variability" as an architectural key concept and first-class quality attribute.

## Categories and Subject Descriptors
D.2.10 [**Software Engineering**]: Design – *methodologies, representation;* D.2.11 [**Software Engineering**]: Software Architectures – *data abstraction, languages;* K.6.3 [**Management of Computing and Information Systems**]: Software Management – *software development*

## General Terms
Management, Documentation, Design, Theory.

## Keywords
Software architecture, variability, product lines, questionnaire, mini focus group.


*VaMoS '11*, January 27–29, 2011, Namur, Belgium.


## 1. INTRODUCTION

### 1.1 Problem Statement and Related Work
Supporting variability in software systems is essential to manage commonalities and differences across software, and to accommodate software reuse in different organizations and product versions. Systematically identifying and appropriately managing variability among different systems distinguishes variability from other approaches that support reuse [7]. The main cause for variability is the delay of design decisions to the latest point that is economically feasible. Examples of mechanisms to accommodate variability include software product lines, configuration wizards and tools in commercial software, configuration interfaces of software components in component-based software engineering, or the dynamic runtime composition of web services [29].

So far, variability has primarily been studied in the software product line domain [7-10]. In general, variability in the product line domain is understood as the ability of an (software) artifact to be configured, customized, extended, or changed for a specific context, in a preplanned manner [3]. Moreover, most definitions of variability in the product line domain include the concepts of "variation point", "variant" and "core asset" [7, 11]. This means, variability is often understood as "anticipated" change, i.e., change that is mostly foreseen, with predefined points of potential change and adaptation, as well as options for how to adapt software systems. In addition, there is evolution of variability which might not necessarily be anticipated.

The product line community also introduced the notion of "product line architecture" [1]: The product line architecture describes concepts and structures to achieve variation in features of different products, while sharing as many parts as possible in the implementation [17]. Thus, the product line architecture captures the central design of all products of the product line, including variability and commonalities of several product instances [30].

However, compared to traditional software architecture [14] (please note that we use singular when referring to the discipline of "software architecture"), product line architectures have a reduced scope with regard to variability. First, product line architectures address variability explicitly and have a limited focus by emphasizing "features", "variation points", "variants", etc. This means, in product lines, variability is captured in features and decisions. On the other hand, variability in the context of

software architecture is treated as a quality attribute and a cross-cutting concern. Software architecture considers variability in a broader scope and acknowledges that variability is a concern of different stakeholders, and in turn affects other concerns. As argued by Bachmann and Bass, variations in a software architecture must be made explicit [2].

Second, product line architectures encompass limited conceptual models, such as feature models or decision models and focus on component-and-connector models [5]. However, variability in other architectural models or views that are particularly relevant for software architecture (deployment models, information models, development models) has not yet been addressed sufficiently. This is particularly true for the impact of preplanned change on quality attributes.

Third, a product line architecture assumes the existence of a product line infrastructure, including related processes (such as core asset development, product development, management) [11]. This is rarely the case for software architectures which are subject to variability. As recently argued by Hilliard, variability is a key fact of "most, if not all, systems" and therefore a relevant concern for the architectures of those systems [13]. This means, variability is not limited to product line architectures but is widespread. Software architects encounter many situations where variability occurs and must be handled. These situations occur due to similar reasons as in product lines and include: deferral of design and implementation decisions and the resulting choices among one or more alternatives; configuration of single systems for customization; multiple deployment; operation and / or maintenance scenarios; planned evolution of a system over its life cycle, self-* (-adaptive, -healing, - managing, etc.) systems, to achieve system qualities such as adaptability, etc. [13].

As with many system properties, identifying and managing variability of a system (either single systems, product lines, system of systems, etc.) early on is preferred over discovering and addressing variability later in the life cycle [28]. As variability is pervasive [13], software architects should be given proper support for dealing with it. It is essential for the architect to have suitable tools for representing, managing and reasoning about variability. However, to provide support for variability, an understanding of variability in the context of software architecture has to be gained first. Compared to the product line domain, no common definition for variability exists in the software architecture domain. Therefore, the overall question that we address in this paper is *how variability is understood in the context of software architecture*. More specific questions will be outlined in Section 2.3.

## 1.2 Goals and Contributions

The goal of this paper is to report the results of a study that aimed at obtaining a better understanding of variability in the context of software architecture. For that reason, we provide our observations from conducting a survey (using questionnaires) and a mini focus group to collect information from experts. Our findings provide an insight into the differences that exist in the notion of variability in the product line domain versus variability in software architecture. Moreover, our observations might be used to formulate hypotheses for future studies. Furthermore, our observations could act as input for further discussions about variability in the context of software architecture. In particular, current software architecture description methods do not provide extensive support for variability. By expanding architecture descriptions with new stakeholders, concerns, models, etc. we can provide architects with tools for representing, managing and reasoning about variability in software architectures.

## 1.3 Paper Structure

In Section 2 of this paper we discuss the design of the study, including the participants and data collection. The results of the questionnaire-based survey are discussed in Section 3. The results of the mini focus group can be found in Section 4. Section 5 lists limitations of our study. Finally, Section 6 concludes this paper.

# 2. RESEARCH METHOD

The study consisted of 2 parts: a questionnaire-based survey and a mini (pseudo) focus group [25]. We use the term "pseudo" as we did not strictly follow all guidelines for the focus group research method [18, 23]: We did not prescribe a predefined list of topics to discuss but led the discussion in the focus group around the overall question of the study described in Section 1.1. Moreover, the focus group included only 4 participants, rather than 6 to 12 as often recommended (hence, "mini" focus group). Both, the survey as well as the focus group were conducted in the context of a workshop at the "European Conference on Software Architecture" (ECSA) in August 2010. Conducting the study at ECSA allowed us to apply purposive sampling when selecting subjects: We targeted subjects from the architecture community, with a background in software architecture and an interest in variability.

## 2.1 Design of the Survey

The survey was planned as an exploratory survey. The data was collected using a paper-based questionnaire which was group-administered (i.e., researchers were available to answer any questions). This was to mitigate the risk of ambiguous or poorly understood questions. The questionnaire included 8 open questions which resulted in qualitative data, and 2 questions for which predefined ratings could be provided. The questions included in the questionnaire will be outlined and motivated in Section 2.3. The survey was started during a break of the workshop. Participants were asked to return the questionnaire whenever they felt ready. In total, 26 questionnaires were handed out, with 11 questionnaires being returned (i.e., 42% response rate).

## 2.2 Participants of the Survey

To get meaningful data, participants must a) have an interest in variability in software architectures, b) possess knowledge and expertise in variability in software architecture, and c) be willing to share their knowledge. Therefore, we decided to apply purposive sampling to recruit participants and conducted the study in the context of a workshop on variability at a premium software architecture conference (ECSA 2010).

Demographic information about participants can be found in Table 1 (please note that "SE" in the first row of Table 1 stands for "Software Engineering" and "SA" stands for "Software Architecture", respectively). Information and background of participants was collected as part of the questionnaire. Experience in software engineering, software architecture and variability includes working experience as well as experience from researching the topics. On the questionnaire, participants P4 and

P7 indicated no extensive "hands-on" experience, but gained their knowledge mainly through research activities. The remaining participants labeled as "Researcher" indicated some industrial experience on the topics, either from performing industrial research, or from previously working in industry.

**Table 1. Demographic information about participants**

| # | Experience (years) | | | Country | Role |
|---|---|---|---|---|---|
| | SE | SA | Variability | | |
| P1 | 9 | 5 | 3 | Belgium | Industrial researcher |
| P2 | 8 | 6 | 3 | USA | Researcher |
| P3 | 5 | 2 | 2 | Finland | Project manager |
| P4 | 5 | 2 | 1 | Sweden | Researcher |
| P5 | 20 | 8 | 5 | Denmark | Software architect |
| P6 | 10 | 10 | 4 | Spain | Industrial researcher |
| P7 | 5 | 2 | 2 | Spain | Researcher |
| P8 | 10 | 1 | 2 | Netherlands | Researcher |
| P9 | 8 | 3 | 3 | Netherlands | Researcher |
| P10 | 20 | 15 | 15 | Sweden | Researcher |
| P11 | 8 | 5 | 2 | Belgium | Researcher |

## 2.3 Survey Questions

In the following, we will outline and motivate the 10 questions of the questionnaire. We split the questions into three groups which all relate to the goal of gaining an understanding of variability in software architecture: 1) general questions about variability, 2) questions about challenges imposed by variability, and 3) questions about approaches to address variability in software architecture (see Figure 1). The groups of questions provide a consecutive line of reasoning.

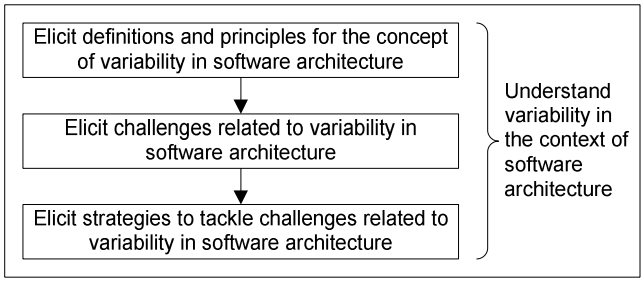


**Figure 1. Line of reasoning of survey questions**

### *2.3.1 General Questions*

General questions aim at understanding the principles of variability in software architecture and basic definitions that participants have about variability. Understanding the principles is the foundation for identifying challenges related to variability.

**Question 1:** We were interested in whether or not the software architecture domain follows the same understanding and applies the same definitions for variability as the product line domain. This resulted in the first question: *What is your working definition of variability in the context of software architectures*? Answering this question helps get an idea of how the software architecture community understands variability. Based on this understanding, we can identify which methods or approaches we can use to tackle challenges related to variability in architecture.

**Question 2:** Many concepts and theories exist in software engineering. However, many ideas are not applied in practice as they do not address a significant problem or lack usability due to poor tool support [24]. This might also be the case for variability in software architecture. To avoid the development of unnecessary new theories, concepts and methods about variability in software architectures, the following question was stated: *Based on your experience, is support for "handling" variability mainly an issue of better tool support rather than new concepts, theories, methods or techniques?* Please note that we use the expression "handling" variability rather than "managing" variability (as commonly used in the product line domain). As argued by Svahnberg et al., managing variability is only one of several activities in the context of variability (besides for example, introducing and implementing variability) [27]. This was also communicated to the subjects that participated in the survey.

**Question 3:** In product line engineering, many types of variability exist, defined in different dimensions [22, 26]. For example, a variation point could be open or closed, mandatory or optional. Moreover, variability could be resolved at runtime, or design time. For handling and managing variability in the architecture, it is essential to have an understanding of what variability might occur at the architectural level. Thus, we asked participants about the types of variability they identify in the context of software architecture. Please note that we did not define "type" but were interested in any kind of categorization or differentiation of variability that participants could identify: *Based on your experience, what "types" of variability occur in software architectures*?

### *2.3.2 Challenges in Variability in Architectures*

This group of questions aims at identifying challenges related to variability in architecture. Answering these questions provides a) a rationale for handling variability in architectures (i.e., if there were no (new) challenges, we would not need any methods to handle variability), and b) a basis for developing methods for handling variability in software architectures.

**Question 4:** In the context of software product lines, challenges have been identified [8, 9]. If variability in software architectures would impose the same or similar challenges, we might be able to apply the same or similar strategies as in the product line domain to software architectures to cope with these challenges. Therefore, we formulated our fourth question: *What do you think are the biggest challenges in "handling" variability in software architectures*?

**Question 5:** The software architecture is, more than any other artifact during software development, concerned with ensuring

that quality attributes can be achieved. In the context of software architecture, architecture quality attributes play a significant role as they act as key drivers for designing systems. Therefore, we were interested in how participants perceive the relationship between variability and quality attributes: *Based on your experience, is the relationship between architecture quality attributes and variability a concern that needs special attention*? Answering this question helps set the right focus on how to relate variability and quality attributes.

**Question 6:** Many new architecture paradigms are currently emerging (such as cloud computing or service-oriented computing). Such paradigms might impose new constraints on handling variability in software architecture. Thus, we specifically asked participants about their opinion on these issues with regard to methods to address variability: *Based on your experience, is there any difference (with regard to concerns, techniques, models, etc.) in variability issues in emerging architecture paradigms*?

### *2.3.3 Strategies to Tackle Variability-related Challenges*

This group of questions aims at identifying methods or techniques that could help tackle challenges related to variability in software architecture.

**Question 7:** As a follow-up question to Question 4, we asked the following: *Based on your experience, what has been the most promising action to tackle the challenges*?

**Question 8:** As mentioned earlier, concepts from the product line domain might be reused for addressing variability in software architectures. To get an insight into how architects judge the need to develop new methods beyond product lines, Question 8 was formulated. Question 8 aimed at identifying the potential for addressing variability by applying methods, techniques, etc. used or developed outside the product line domain: *Is there a need to address variability in software architectures beyond the product line domain*?

**Question 9:** Reference architectures are a core element of product lines [21] and help cope with variability. Reference architectures are created by capturing the essentials of architectures and by taking into account future needs. Reference architectures consider variability to provide guidance when developing architectures for new systems, new versions or extensions of product families [12]. Therefore, we asked Question 9: *How important do you rate reference architectures for managing variability*? For this question, we asked participants to rate the importance on a 6-point scale: -3 = "totally irrelevant", -2 = "unimportant", -1 = "somewhat unimportant", 1 = "somewhat important", 2 = "important", 3 = "very important".

**Question 10:** In the software architecture domain, architecture views and viewpoints have become an integrated part of architecture descriptions [16]. Viewpoints describe architectures from the perspective of particular stakeholders and focus on specific concerns. One concern could be variability. Therefore, we formulated the last question as follows: *Below* [Table 2] *is a list of potential requirements for variability viewpoints. Based on your experience, please rate each requirement for a variability viewpoint.* Subjects could assign values between 1 (least important) and 10 (most important) to each requirement. Please note that Table 2 includes 6 potential requirements as provided by us. We selected these requirements based on our understanding of the potentially most useful requirements. As we cannot claim that this selection is exhaustive neither valid for all situations, we also allowed subjects to provide their own requirements and add them to the list. However, no participant added any requirement.

**Table 2. Potential requirements for a variability viewpoint**

| # | Description |
|---|---|
| Req 1 | It must describe where in the architecture variability occurs. |
| Req 2 | It must support consistency management to ensure consistency with other viewpoints (or between views). |
| Req 3 | It must support management of dependencies between variation points and between variants represented within a view. |
| Req 4 | It must support different "types" of variation points (see also Question 3). |
| Req 5 | It must allow the representation of different times of resolving variability. |
| Req 6 | It should be part of a variability viewpoint catalogue, as one variability viewpoint might not be enough to describe all relevant concerns. |

## 2.4 Design of Focus Group

The focus group was scheduled for 2 hours. The 4 participants were a subset of the group that participated in the survey (namely P1, P2, P9 and P10). Participants were chosen based on their interest in joining the focus group. The mini focus group was conducted at the end of the workshop. Rather than recording the focus group session, manual notes were taken by one of the researchers. The flow of the focus group was designed around the problem of variability in software architectures. No specific topics had been predefined, i.e., no specific sequence of topics was followed.

## 2.5 Data Analysis

The data collected in the survey was analyzed as discussed in the subsection of Section 3. The data collected in the mini focus group was analyzed as outlined in Section 4.

# 3. RESULTS OF THE SURVEY

In this section, we present the results of analyzing the data collected in the survey. However, rather than presenting conclusive findings or empirical evidence, we present a set of observations that we made from analyzing the data. Please note that we do not present the answers to questions in the same order as presenting the questions in the previous section. This is because we grouped the answers based on topics that emerged from the answers. These topics include definitions of variability (Section 3.1), challenges with variability in software architectures (Section 3.2), ways to address challenges (Section 3.3), new theories versus better tools (Section 3.4), emerging architecture disciplines (Section 3.5), and variability viewpoints (Section 3.6).

## 3.1 Definitions of Variability

We asked participants for their working definition of variability (see Question 1 in Section 2.3). Interestingly, none of the 11 working definitions that we collected included the concepts of "variation points", or "version". Only the definition given by P2 included the concept of "variants". Therefore, we performed a frequency analysis of terms used in the 11 working definitions to identify the concepts that participants considered important. The results are shown in Table 3. When creating Table 3, we removed filling terms, such as "a", "and", as well as generic terms, such as "software" or "systems". Please note that in Table 3 similar terms have been grouped if used with the same meaning (for example, "property" and "ability"). "% (1)" in Table 3 denotes the relative occurrence of a term with regard to the total number of occurrences of terms in Table 3. "% (2)" in Table 3 denotes the relative occurrence of a term in the 11 working definitions. From Table 3 we see that the most important characteristic of variability seems to be that it is a property or aspect of architectures that includes differences and results in changes in the architecture. These characteristics do not specify if this change is "pre-planned" or occurs in the context of reuse.

**Table 3. Frequency analysis of terms used for defining variability**

| Terms | Frequency | % (1) | % (2) |
|---|---|---|---|
| different / differences / differ | 5 | 13 | 45 |
| change / changing / changes / adapting | 5 | 13 | 45 |
| property / attribute / aspect / ability | 5 | 13 | 27 |
| quality | 2 | 5 | 27 |
| factors | 2 | 5 | 18 |
| behavior | 2 | 5 | 18 |
| vary / flexible | 2 | 5 | 18 |
| significant / influential | 2 | 5 | 18 |
| requirements / features | 2 | 5 | 18 |
| reconfigure / customized | 2 | 5 | 18 |
| variants | 1 | 3 | 9 |
| structure | 1 | 3 | 9 |
| SPL | 1 | 3 | 9 |
| specification | 1 | 3 | 9 |
| reusing | 1 | 3 | 9 |
| extend | 1 | 3 | 9 |
| context | 1 | 3 | 9 |

Next, we aggregated all 11 definitions to formulate one distinct definition (using reciprocal translation [20]). However, due to the different notions that participants used to describe variability, we created four different definitions, each with a distinct focus:

The first definition (D1) is similar to definitions that can be found in the product line domain [7-10]: Variability in software architectures is the need of software to accommodate change and the ability to create architectures for different products. This includes reconfiguring the architectural structure and behavior in an efficient and effective manner.

The second definition (D2) is similar to the classification of variability that can be found in software product quality models that define variability as a subcategory of maintainability and changeability [6, 15, 19]: Variability in software architectures describes how well an architecture supports flexibility in a certain aspect, with an exact specification of the differences.

The third definition (D3) describes variability in terms of how variability is achieved: Variability in software architectures is achieved through different variants that exhibit different behavior, and the determination of different features, whilst reusing common artifacts, without changing the scope of the architecture.

The fourth definition (D4) treats variability as an architecture design time quality attribute [4]: Variability in software architectures is the ability of influential factors on design to vary. In other words, variability is a quality attribute whose solution impacts other quality attributes (causing trade-offs between variability and other quality attributes).

The aggregated definitions are very generic and thus might indicate a lack of formal semantics for variability in architectures.

**Observation 1:** There is no commonly agreed definition of variability in the domain of software architecture. Furthermore, the definition of variability in software architecture seems to differ from the concept of variability in product lines. This might mean that new methods for handling variability in architectures are needed, rather than merely applying well-known ideas from product lines.

## 3.2 Challenges with Variability in Software Architecture

Question 4 aimed at identifying challenges in the context of variability in software architectures. Subjects identified challenges based on their individual perceptions and definitions about variability. Based on the responses from the subjects, we used "line of argument synthesis" to identify the "main theme" of different challenges [20]. This resulted in 3 groups of challenges related to variability in architectures:

**Complexity:** This includes the ability to keep variability simple and straight-forward, the identification of what to change and what to vary, the identification of how to change and how to vary, as well as when.

**Formality:** This includes the lack of formal, precise, and rigorous definitions of underlying variation relations, as well as the validation of variability models.

**Management:** This group includes many different things: updating architectures (and managing traceability between updates), managing the dependencies of variations to hardware, dynamic variability, traceability of variability from feature models to architecture model, as well as consistency management and evolution.

**Observation 2:** As can be seen in the above list, challenges named by participants are similar to known challenges in the context of architectures (traceability, consistency management, etc.). Moreover, the challenges are similar to challenges found by Chen and Babar in the context of variability management in product lines (e.g., complexity and management) [8].

## 3.3 Ways to Address Challenges

Based on the challenges identified through Question 4, we collected ways to tackle these (Question 7). Participants named using formal semantics for modeling and analysis, patterns, different architectural views, stepwise refinement and separation of concerns as potential counter measures.

Interestingly, the definition and use of reference architectures as a method to respond to challenges related to variability in architectures was considered as significant (Question 9). 7 out of 11 subjects considered reference architectures as important and 2 subjects considered them as somewhat important. 1 subject considered them as unimportant and 1 subject voted for somewhat unimportant. Please note that we could not identify any correlation between rating the importance of reference architectures and the experience of subjects.

**Observation 3:** Challenges related to variability might be tackled by already known approaches. Reference architectures are considered as significant help in tackling variability concerns.

## 3.4 New Theories versus Better Tools

We also asked participants if handling variability is more an issue of better tools, or if we need new theories, concepts, methods or techniques (Question 2). Interestingly, 9 of 11 participants indicated a need for more theories, new concepts and a better understanding of the concept, rather than the development of tools based on existing methods and ideas. As stated by P2, new and better tools would be the result of new and better concepts. 2 subjects (P4 and P7) did not have an opinion about this issue. P4 and P7 were the subjects with no "hands-on" experience and least background (see Section 2.2).

In particular, the relationship between variability in architectures and the impact on quality attributes was considered as a major challenge that would require new theories, as stated by 8 of 11 subjects (Question 5). 3 subjects (P4, P5 and P8) indicated that they had not enough experience to properly answer this question.

Moreover, the complexity of many different types of variability was seen as a source for problems (Question 3). As subject P5 stated, there are far too many types of variability as "the software architecture is constantly under attack from the desire to quickly make the next variant". Overall, subjects listed the following types: a) status (open, closed), b) binding time, c) realization technique, d) functional and non-functional, e) cross-cutting and non-cross-cutting, f) structural (static) / behavioral (dynamic) / data, g) platform / product (different product / within product). Therefore, theories and concepts to formalize variability types in software architectures would be needed.

When asked about new methods or theories beyond the product line domain (Question 8), 8 out of 11 subjects indicated a need for extending the concept of variability beyond product lines. The remaining subjects (P1, P4, and P8) did not see any need for investigating variability beyond product lines. In particular, more interaction with process models and formal methods was considered necessary (P2). Also, methods beyond product lines would need to address flexibility, evolution, reusability and quality (P6), or autonomic computing, i.e., autonomic product configuration based on monitoring and goal models (P10). Please note that when relating the experience of subjects and their judgment about the need for new methods, we could not find a strong correlation.

**Observation 4:** New theories with regard to variability in architectures are needed. Moreover, the relationship between variability and quality attributes needs to be explored further. To achieve this, variability might be extended beyond the product line domain.

## 3.5 Emerging Architecture Disciplines

We were interested in how participants judge variability in the context of emerging architecture disciplines (Question 6). Subjects mentioned that there might be new types of variability and relationships (P8: less control in service-based systems and more interface-oriented, thus increasing the degree of complexity). Overall, subjects did not indicate any strong opinion on this topic. However, P1 stated that different architecture paradigms would support different variability mechanisms.

**Observation 5:** Whether or not emerging architecture paradigms require special treatment with regard to variability is not clear.

## 3.6 Variability Viewpoints

One way of addressing the complexity of architectures is to create proper architecture descriptions. Parts of such descriptions are architecture viewpoints and architectural views which describe specific concerns of a system. Therefore, we asked subjects to rate certain requirements for a variability viewpoint and also if they have their own requirements for a variability viewpoint (Question 10). The results are shown in Figure 2.

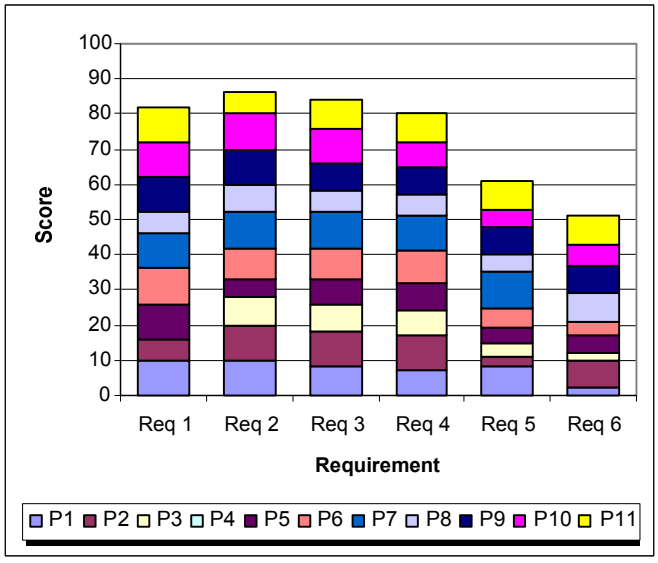


**Figure 2. Requirements for variability viewpoints.**

The requirements listed in Figure 2 correspond to requirements listed in Table 2. The coloring per subject and requirement indicates the importance that a subject assigned to a requirement. For example, P1 assigned a high importance to Req 1, but a very low importance to Req 6.

**Observation 6:** Consistency management seems to be the requirement with the highest priority for a variability viewpoint. Creating a variability viewpoint catalogue seems to be least important.

# 4. RESULTS OF THE MINI FOCUS GROUP

The mini focus group also aimed at understanding variability in the context of software architectures. In this section, we relate the focus group discussion to the observations we made from analyzing the questionnaires.

Most of the discussion time in the focus group (about 1 hour) was spent on defining variability as such. This is an indicator that no clear understanding of variability exists (see also Observation 1 in the previous section). Questions such as if variability itself is a quality attribute, or (how) does variability impact quality attributes were discussed. For a more thorough analysis, we studied the transcripts of the focus group and grouped the discussion topics around the following issues:

**Nature of variability:** Two group participants argued for defining variability in terms of the very basics, i.e., variability as the difference / similarity between two or more products. On the other hand, there was an agreement that variability is a means to achieve quality attributes. Moreover, variability was considered as a quality attribute itself which impacts the architecture. As one participant stated, variability might not be just a "normal" quality attribute, with regard to functionalities and other qualities, but a quality that impacts other quality attributes and needs to be specified explicitly. This confirms Observation 1 as well as Definition D4 discussed in the previous section. However, we believe that the understanding of variability as a quality attribute that affects other quality attributes is misleading and might indicate a common misunderstanding in the architecture community. In fact, we think that it is not a quality attribute itself that impacts other quality attributes, but it is the architecture measures taken to achieve a quality attribute that impact other quality attributes.

**How does variability "fit in":** There was a common understanding that variability can occur in time and space, within a product or across products; it needs to be defined in terms of "where" and "when" (binding time). This understanding complies with the understanding of variability in product lines. However, as argued by participants, from the software architecture perspective variability exists beyond the product line domain and is architecture-driven (this confirms Observation 4 discussed in the previous section).

**Why utilize variability:** According to all participants, variability helps support a number of decisions, but is influenced by a number of factors (such as interpretation, uncertainties, market). Moreover, it allows deferring decisions and the evaluation of conditions. This is similar to the understanding that can be found in the product line domain. However, in the focus group there was an emphasis on the relation and impact that variability has on quality attributes (in particular, to use variability to "control" quality attributes), which seems to be not the case in the product line domain.

**Trade-offs involved:** As argued by 3 participants, variability can occur in a single quality attribute, with different quality levels (e.g., different settings for performance), or in multiple quality attributes (which might lead to a restriction of the architecture design). On the other hand, there is a trade-off between quality and functionality. Interestingly, the interplay between variability and quality attributes, and solutions to achieve variability (e.g., tactics, patterns) was not brought up by any of the participants.

The conclusions of the focus group were as follows: First, there was a large diversity in the different meanings of variability. This also confirms Observation 1 made from the survey. Second, in general, variability is a means for handling "differences". However, no details were provided on how to scope "differences" (e.g., with regard to functionality, quality, or in terms of a single product, multiple products, etc). Third, variability is related to functionality and quality attributes (such as performance) as well as "intrinsic" architecture quality attributes (i.e., design time quality attributes, such as modularity), see Figure 3. The arrows indicate interdependencies between variability, quality attributes, functionality, and intrinsic architecture quality attributes. However, as mentioned when discussing the "Nature of variability", based on the understanding in the software architecture domain we believe that it is not the quality attribute itself that impacts other quality attributes.

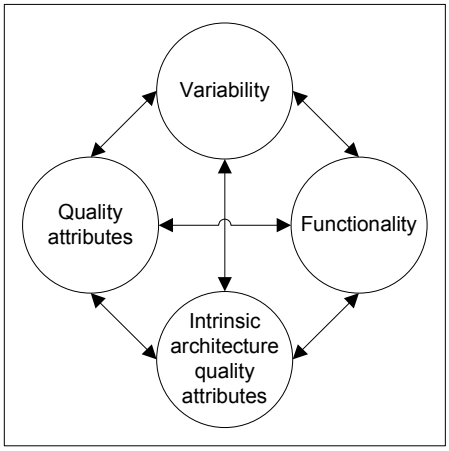


**Figure 3. "Context" of variability in software architecture**

# 5. LIMITATIONS

The most significant limitation of this study is its statistical significance. By no means have we claimed statistical relevance of our findings. On the contrary, we see the presented work as an exploratory insight into the understanding of variability in the software architecture domain, in contrast to the well-known understanding of variability in the software product line domain. In detail, the limitations are as follows:

First, we conducted the study at an academic venue. As a consequence, most participants of the survey and the mini focus group came from academia. This might impose limitations with

regard to the practical relevance of our results. For example, we concluded that new theoretical concepts are needed rather than new tools. This observation might be biased by the fact that we collected data at a conference, rather than a trade fair or any other practitioner-oriented venue. However, as can be seen in Table 1, most participants had significant experience, some even in an industrial context.

Second, the number of participants was limited. Only 11 subjects participated in the survey and 4 participants in the mini focus group. However, we needed subjects with a particular background and expertise, and a serious interest in the topic. Therefore, the scope of potential participants was limited by the definition of our research problem.

Third, the focus group only describes the participants' personal knowledge and beliefs about variability in software architectures. This might have led to inaccurate responses.

Fourth, some of the data analyses might have been subjective. In particular, the analysis of the focus group should be understood with caution.

Fifth, many definitions for software architecture exist. Thus, it could be argued that, as a consequence, many definitions for variability in software architecture exist. However, we were interested in getting a first insight into what these definitions (and the related understanding) could be. In particular, several questions used in the survey depend on the definition of variability. Again, in our study this was desirable as it gave us the chance to obtain different perceptions and ideas.

# 6. CONCLUSIONS

The observations from this study provide useful information about the understanding of variability in the context of software architectures. Some of our observations and in particular the identified challenges confirm findings from the product line domain (e.g., Chen and Babar [8] or Bosch et al. [7]). However, our observations suggest that there is no common understanding of the nature of variability in the software architecture community, in contrast to the product line domain, where a clear understanding of variability exists. This gap in understanding seems to be neglected by variability researchers.

One future direction of our work is towards variability viewpoints as part of architectural descriptions. We are investigating how viewpoints and views can be used to support the description and reasoning about variability in software architectures. Different stakeholders typically have different concerns with regard to variability. Usually, only a part of the whole variability concern is of interest for a particular stakeholder.

# 7. ACKNOWLEDGMENTS

The authors thank the subjects of the survey and the participants of the focus group. The authors also thank the anonymous reviewers who helped improve the paper. This work has been partially sponsored by the Dutch Joint Academic and Commercial Quality Research & Development (Jacquard) program on Software Engineering Research via contract 638.001.207 SAS-LEG: Software As Service for the varying needs of Local EGovernments.

# 8. REFERENCES

[1] Ahmed, F. and Capretz, L.F. 2008. The Software Product Line Architecture: An Empirical Investigation of Key Process Activities. *Information and Software Technology* 50, 11, 1098-1113.

[2] Bachmann, F. and Bass, L. 2001. Managing Variability in Software Architectures. In *2001 Symposium on Software Reusability* (Toronto, ON, Canada). ACM Press, 126-132.

[3] Bachmann, F. and Clements, P.C. 2005 *Variability in Software Product Lines*, SEI CMU, Pittsburgh, PA.

[4] Bass, L., Clements, P. and Kazman, R. 2003. *Software Architecture in Practice*. Addison-Wesley, Boston, MA.

[5] Bayer, J., Flege, O., Knauber, P., Laqua, R., Muthig, D., Schmid, K., Widen, T. and DeBaud, J.-M. 1999. PuLSE: A Methodology to Develop Software Product Lines. In *Symposium on Software Reusability* (Los Angeles, CA). ACM, 122-131.

[6] Bode, S. and Riebisch, M. 2010. Impact Evaluation for Quality-Oriented Architectural Decisions Regarding Evolvability. In *4th European Conference on Software Architecture* (Copenhagen, Denmark). Springer Verlag, 182-197.

[7] Bosch, J., Florijn, G., Greefhorst, D., Kuusela, J., Obbink, J.H. and Pohl, K. 2002. Variability Issues in Software Product Lines. In *4th International Workshop on Software Product Family Engineering* (Bilbao, Spain). Springer Verlag, 303-338.

[8] Chen, L. and Babar, M.A. 2010. Variability Management in Software Product Lines: An Investigation of Contemporary Industrial Challenges. In *14th International Software Product Line Conference* (Jeju Island, South Korea). Springer Verlag, 1-15.

[9] Chen, L., Babar, M.A. and Ali, N. 2009. Variability Management in Software Product Lines: A Systematic Review. In *13th International Software Product Line Conference (SPLC)* (San Francisco, CA). Carnegie Mellon University, 81-90.

[10] Chen, L., Babar, M.A. and Cawley, C. 2009. A Status Report on the Evaluation of Variability Management Approaches. In *13th International Conference on Evaluation and Assessment in Software Engineering (EASE)* (Durham, UK). BCS, 1-10.

[11] Clements, P. and Northrop, L. 2001. *Software Product Lines - Practices and Patterns*. Addison-Wesley, Boston, MA.

[12] Cloutier, R., Muller, G., Verma, D., Nilchiani, R., Hole, E. and Bone, M. 2010. The Concept of Reference Architectures. *Systems Engineering* 13, 1, 14-27.

[13] Hilliard, R. 2010. On Representing Variation. In *Workshop on Variability in Software Product Line Architectures* (Copenhagen, Denmark). ACM, 312-315.

[14] IEEE Computer Society Software Engineering Standards Committee. 2000 *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*.

[15] ISO/IEC. 2001 *Software engineering - Product quality - Part 1: Quality model*, Geneva, Switzerland.

[16] ISO/IEC. 2010 *Systems and Software Engineering - Architecture Description*, Geneva, Switzerland.

[17] Jazayeri, M., van der Linden, F. and Ran, A. 2000. *Software Architecture for Product Families: Principles and Practice*. Addison-Wesley, Reading, MA.

[18] Kontio, J., Lehtola, L. and Bragge, J. 2004. Using the Focus Group Method in Software Engineering: Obtaining Practitioner and User Experiences. In *International Symposium on Empirical Software Engineering* (Redondo Beach, CA). IEEE Computer Society, 271-280.

[19] Mari, M. and Eila, N. 2003. The Impact of Maintainability on Component-based Software Systems. In *29th Euromicro Conference* (Belek-Antalya, Turkey). IEEE Computer Society, 25-32.

[20] Noblit, G.W. and Hare, R.D. 1988. *Meta-Ethnography: Synthesizing Qualitative Studies*. Sage Publications, Newbury Park, CA.

[21] Pohl, K., Boeckle, G. and van der Linden, F. 2005. *Software Product Line Engineering - Foundations, Principles, and Techniques*. Springer Verlag, Berlin / Heidelberg.

[22] Reiser, M.-O., Kolagari, R.T. and Weber, M. 2007. Manifoldness of Variability Modeling - Considering the Potential for Further Integration. In *Second IFIP TC 2 Central and East European Conference on Software Engineering Techniques* (Poznan, Poland). Springer Verlag, 291-303.

[23] Seaman, C.B. 1999. Qualitative Methods in Empirical Studies of Software Engineering. *IEEE Transactions on Software Engineering* 25, 4, 557-572.

[24] Shaw, M. 2005. Sparking Research Ideas from the Friction Between Doctrine and Reality. In *5th Working IEEE/IFIP Conference on Software Architecture* (Pittsburgh, PA). IEEE Computer Society, 11-16.

[25] Singer, J., Sim, S.E. and Lethbridge, T.C. 2008. Software Engineering Data Collection for Field Studies In *Guide to Advanced Empirical Software Engineering*, Shull, F., Singer, J. and Sjoberg, D., Eds. Springer Verlag, Berlin / Heidelberg, 9-34.

[26] Sinnema, M. and Deelstra, S. 2007. Classifying Variability Modeling Techniques. *Information and Software Technology* 49, 7, 717-739.

[27] Svahnberg, M., van Grup, J. and Bosch, J. 2005. A Taxonomy of Variability Realization Techniques. *Software - Practice and Experience* 35, 8, 705-754.

[28] Thiel, S. and Hein, A. 2002. Modeling and Using Product Line Variability in Automotive Systems. *IEEE Software* 19, 4, 66-72.

[29] van Gurp, J. and Bosch, J., Eds. 2003. *Proceedings of the Software Variability Management Workshop*, Groningen, The Netherlands.

[30] Verlage, M. and Kiesgen, T. 2005. Five Years of Product Line Engineering in a Small Company. In *27th International Conference on Software Engineering* (St. Louis, MO). ACM, 534-543.